BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

As rescanning documents *will not* correct images, please do not report the images to the Image Problem Mailbox.

An Architecture and API for of Transport and Upper Layer Protocol Processing Acceleration

I. DESCRIPTION

A. Related Applications

[01] This Application claims priority from co-pending U.S. Provisional Application Serial No. 60/446,558 filed February 12, 2003, the contents of which are incorporated herein by reference.

B. Field

[02] This disclosure teaches techniques generally related to the handling of transport and upper layer protocol connections by an intelligent network interface card (NIC). More specifically, the acceleration of NIC by restructuring the partitioning of the network layer model is also discussed.

II. Background

[03] Network communications often use a seven layer communication model developed by the International Standards Organization (ISO) and known as the Open Systems Interconnection (OSI) networking suite. The seven layers are physical (L1), logical or data link (L2), network (L3), transport (L4), session (L5), presentation (L6), and application (L7). Each layer performs certain functions. When all seven layers work together, data is brought to the application level if received from the network. Data from the application level is also sent down the model and transferred over the physical transportation media.

[04] In the physical layer (L1), the physical properties of the various communications media, as well as the electrical properties and interpretation of the exchanged signals are handled. The logical layer (L2) handles the logical organization of data bits transmitted through a particular medium. The network layer (L3) describes how a series of exchanges over various data links can deliver data between any two nodes in a network, for example Internet protocol (IP). The transport layer (L4) handles the quality and nature of the data delivery, for example, transport control protocol. The session layer (L5) handles the organization of data sequences larger than the packets handled by lower layers. The presentation layer (L6) describes the syntax of data being transferred, and in the application layer (L7) the actual implementation of the application gets performed, for example, an implementation of a file transfer protocol (FTP).

[05] Traditionally L1 and L2 were implemented in hardware and L3 through L7 in software, mostly in a host or a client. In addition to the seven-layer model, OSI also includes a set of protocols for handling data.

[06] This model is now widely used as an operating model for transporting packetized data. While acceleration of the operations in a network are required, the inherent architecture of the network is maintained to enable easy configuration of products from a multitude of suppliers, all generally conforming to the standard.

[07] A typical architecture of a network system is shown in Fig. 1. Architecture 100 comprises of an application 110, protocols 120 and 130 and drivers 140 implemented in software, usually on the host, and a network interface card (NIC) 150 which deals with at least layers 1 and 2. Therefore, in this implementation, layers 3 and above are implemented in software on the host. The need for

acceleration of the operations involved in handling of data to be communicated over the network arises from the fact that networks provide significantly faster communication speeds (now approaching over 10 giga bits per second) as well as the volume of data to be transferred. Handling such a volume of data by a host is a daunting task requiring significant computing power not normally available to such hosts.

[08] Solutions in the related art attempt to provide acceleration of the operations involved in handling data communication. These related art solutions generally take one of two routes. The first breaks from the standard OSI model and suggests other solutions to accelerate performance. Examples for such approaches are shown in US patent 6,018,530 by Chakravorty and 6,273,622 by Ben-David.

[09] The second approach attempts to accelerate certain portions of the protocols, for example, implementing a higher layer (for example L3 or L4), that were traditionally handled in software, in hardware. Examples of such approaches are shown in US patents 5,717,691 by Dighe et al. and US patent application 20020087729 by Edgar. However, none of the related art approaches teach architectures and methods for offloading transport protocols and upper layer protocol from the host software.

III.Summary

[10] It would be therefore advantageous to provide an architecture designed to be consistent with the seven layer model but providing for an accelerated performance of the transport protocol and upper layer protocols.

[11] To realize the advantages, there is provided, a network interface card comprising an upper layer protocol (ULP) handler, a TCP handler capable of interfacing with said ULP handler and,

a link handler. The network interface card is adapted to take over and perform at least one session layer function of a host computer connected to a network.

[12] In a specific enhancement, the network interface card is a layer 5 network interface card in a network implementing an open systems interconnection (OSI) protocol.

[13] In another specific enhancement, data from the network is received and processed by the network interface card.

[14] In another specific enhancement said processing comprises taking over and performing at least one function of a physical layer, a data link layer, a network layer or a transport layer from the host computer.

[15] More specifically, the ULP handler is adapted to communicate with a ULP driver of said host computer.

[16] In yet another specific enhancement, the TCP handler is adapted to communicate with a transport accelerator driver of said host computer.

[17] In still another specific enhancement, the TCP handler is adapted to communicate with the network.

[18] In still another specific enhancement, the link handler is adapted to communicate with a link driver of said host computer.

[19] In still another specific enhancement, the link handler is adapted to communicate with a network.

[20] In still another specific enhancement, the network interface card further comprises at least one of a transport accelerator, embedded accelerator, portable stack, embedded link driver and embedded applications.

[21] In still another specific enhancement, the network interface card is capable of receiving commands from an enhanced stack belonging to said host, said enhanced stack being further capable of supporting session layer acceleration.

[22] More specifically, the commands between said enhanced stack and said network interface card are performed using acceleration primitives.

[23] Even more specifically, said network interface card handles only a subset said acceleration primitives sent to said network interface card from a plurality of said acceleration primitives sent to a plurality of network interface card devices.

[24] Even more specifically, at least one of said acceleration primitives is used to establish a direct connection between ULP of said host and said ULP handler.

[25] Even more specifically, said direct connection enables at least one function associated with a TCP/IP layer to be processed on said network interface card.

[26] Even more specifically, said direct connection comprises enables transferring data to said network interface card from said host and transferring data from said network interface card to said host.

[27] Even more specifically, said transferring data to said network interface card includes at least one of a transfer in request, a success transfer in reply and a fail transfer in reply.

[28] Even more specifically, said transferring data from said network interface card includes at least one of a transfer out request, a success transfer out reply and a fail transfer out reply.

[29] Even more specifically said acceleration primitives are enabled by the use of an application programming interface (API) for interfacing between said host and said network interface card, said API being further comprised of a plurality of message primitives.

[30] Another aspect of the disclosed teachings is an API for facilitating interfacing between a host computer and a network interface card, said API including some of the message primitives shown in Fig.10.

[31] Yet another aspect of the disclosed teachings is a method of acceleration of a network operation, said method comprising transferring a connection to a network interface card and taking over from a host computer attached to the network and performing at least one network function at the network interface card.

[32] Still another aspect of the disclosed teachings is a method for acceleration of a session layer network operation, said method comprising sending a sequence of initialization commands from a ULP driver of a host to transport accelerator provider (TAP) of said host. A transfer message is sent from said TAP to a TCP handler of a network interface card (NIC). A synchronization command is sent from the NIC to a server over a network connecting said host computer and said server. A synchronization acknowledgement message is received by the NIC over said network from said server. An acknowledgment message is sent from the NIC to said server. A notification command is sent to a ULP handler of said NIC. A connection notification command is sent from the NIC to said TAP of said host. A connected information command is sent from the NIC to said ULP driver of said host.

IV. Brief Description of the Drawings

[33] The disclosed teachings will become more apparent by describing in detail examples and embodiments thereof with reference to the attached drawings in which:

[34] Figure 1 is a diagram of the traditional layers of TCP based application processing (prior art).

[35] Figure 2 is a diagram of an exemplary implementation of a host offloading of TCP based application processing.

[36] Figure 3 is a diagram of an exemplary implementation of a TCP/ULP acceleration model embodying aspects of the disclosed teachings.

[37] Figure 4 is a diagram of the an exemplary implementation of a connection management for TCP/ULP acceleration model embodying aspects of the disclosed teachings.

[38] Figure 5 is a diagram of an exemplary implementation of a host centric TCP/ULP acceleration model embodying aspects of the disclosed teachings.

[39] Figure 6 is a diagram of an exemplary implementation of an embedded TCP/ULP acceleration model embodying aspects of the disclosed teachings.

[40] Figure 7 is a diagram of an exemplary implementation of connection establishment embodying aspects of the disclosed teachings.

[41] Figure 8 is a diagram of an exemplary implementation of data transmission embodying aspects of the disclosed teachings.

[42] Figure 9 is a diagram of an exemplary implementation of connection closure in embodying aspects of the disclosed teachings.

[43] Figure 10 A-D is a description of messages between a host and the L5NIC.

[44] Figure 11 is a comparison between FreeBSD APIs and the "tl_" in accordance with aspects of the disclosed teachings.

V. Detailed Description

[45] Reference is now made to Fig. 2 where an exemplary implementation of an architecture for acceleration of upper level protocols (ULPs) is shown. In this exemplary implementation, acceleration of TCP/IP is achieved by providing layer 5 network interface card (L5NIC) drivers. These drivers are capable of handling requests directly from a ULP and operating in conjunction with a L5NIC. This implementation minimizes the amount of TCP/IP protocol processing performed by a host and allows the transfer of the raw data from the host to the L5NIC.

[46] Corresponding functionality is also achieved in the opposite direction, i.e., when data is received from a network to be transferred to a host. This data can be processed and received by the L5NIC. Only when ready, this data is provided to the host. This relieves the host from the need to process the requests. ULPs optimized for acceleration may include, but are not limited to: iSCSI Target, iSCSI Initiator, CIF/NSF, RDMA, Generic TCP, etc.

[47] Referring now to Fig. 3, a detailed view of an exemplary implementation of an architecture embodying aspects of the disclosed teachings is provided. A ULP requiring acceleration 310 may provide a request to the enhanced stack supporting acceleration 340. The interface facilitating the receipt and implementation of such requests is described in more detail below.

[48] In addition, Fig. 10A-D, describes in further detail the requests available in an implementation of the disclosed teachings. In Fig. 10, the messages exchanged in the system are described in detail. These messages are useful for offloading the establishment of a connection, processing associated with data transfer associated, and processing associated with connection tear down. The messages are of particular value when used in conjunction with the described L5NIC. The enhanced stack 340 communicates with the L5NIC drivers 350 through a set of generic acceleration primitives further detailed below.

[49] The L5NIC drivers 350 communicate with a specific L5NIC 360 from a plurality of L5NICs, for example L5NIC 360-1 and assign it for the processing of the ULP request made by the ULP requiring acceleration 310. Once a connection is established, the ULP requiring acceleration 310 may communicate directly with the assigned L5NIC, for example L5NIC 360-1, and through ULP driver 320 maintain the connection and operate in an accelerated fashion. Standard applications can continue to operate using the standard model through socket emulation 330 that communicates with the enhanced stack 340. While L5NIC 360 are shown in this implementation to be hardware platforms it is possible to operate with a combination of one or more implementations of L5NIC that are software based.

[50] Reference is now made to Fig. 4 where a schematic diagram of an exemplary implementation of the operation of the enhanced stack 340 with an L5NIC is shown. The diagram should be viewed as an operational diagram rather then a functional description of elements of the solution disclosed herein. It is the task of the enhanced stack 340 to support the transportability of a connection from one accelerator to another as well as the communication with the host.

[51] Specifically, connections are created in the enhanced stack 340 and it controls all the inbound and outbound transfers. Enhanced stack 340 initiates a Transfer_In_Request which is responded to as described in 410. Such a request may either succeed or fail. In case of failure a "fail" message is relayed using Transfer_In_Reply. If the request succeeds, connection is transferred to L5NIC 360.

[52] While in transfer the connection state is frozen. L5NIC 360, to which the connection has been transferred, is now capable of handling the connection. All dynamic context information is maintained by the L5NIC 360. Data will flow directly from the ULP handler (ULP FW) on the L5NIC 360 to the host handling the enhanced stack 340, as described in more detail below. A connection may be ceased by the host through enhanced stack 360 through the use of the Transfer_Out_request which is responded to, as described, in 420. In case of failure, a fail message using Transfer_Out_Reply is sent to L5NIC 360. Otherwise, if the Transfer_Out_ Request is a "success," then the message is sent to the enhanced stack 340 and the connection is moved from the control of the L5NIC 360 back to the host. Detailed examples of establishing a connection, transferring of data, and closing a connection are discussed below.

[53] Fig. 5 shows an exemplary implementation of a generic TCP/ULP acceleration model 500 that is host centric. In this implementation, components ULP stack 510, ULP driver 520, hardware common access 530, enhanced stack 540, transport accelerator 550 and link driver 560, are all implemented in software and are part of a host that is capable of communication with a L5NIC 570. This model can be expanded so that a host could communicate with a plurality of L5NICs.

[54] L5NIC is further connected to the network 580. L5NIC 570 further comprises from a ULP handler 571, TCP handler 572 and link handler 573, capable of communicating with ULP driver 520, transport accelerator 550 and link driver 560 respectively. The TCP handler and the link handler are capable of communicating with a network. Each of the handler units 571, 572, and 573, may be implemented in firmware, hardware, or combination thereof. Connections are initiated by the host and the control of pushing and transferring connections in and out of the L5NIC 570 is part of the tasks assigned to the ULP stack 510.

[55] The enhanced stack is comprised of a portable stack 544 and a transport accelerator provider 542. Portable stack 544 may be based, for example, on FreeBSD 4.5. The enhancements discussed in relation to the disclosed teachings pertain to the transport accelerator provider 542 and include a plurality of acceleration primitives described in more detail below.

[56] The TCP handler and the link handler may be capable of communicating with a network. The L5NIC may comprise at least one of transport accelerator, embedded accelerator, portable stack, embedded link driver and embedded applications.

[57] L5NIC 570 performs function normally related to layers 1 through 5 of the standard communication model like the OSI model. Layer 1 is the actual physical connection. Layer 2 is handled, at least in part, by link handler 573. Layers 3 and 4 are handled, at least in part, by TCP handler 572, and layer 5 is handled, at least in part, by ULP handler 571.

[58] Referring to Fig. 6 a model 600 using an embedded version of an L5NIC 670 is shown. In a system where the embedded version of L5NIC 670 is used, only the

ULP stack 510, ULP driver 520 and the hardware common access 530 are used. All other functions previously handled in software by the host are now embedded as part of the L5NIC 670. Therefore the enhanced stack and the corresponding drivers are now embedded as part of L5NIC 670. The architecture 600 can be modified such that the host can operate without a stack at all. The management of the applications is done through the use of embedded applications 678.

[59] Reference is now made to Fig. 7 where a communication diagram is shown for the purpose of opening a connection between a client and a server using L5NICs on both the client and the server side of the network. The system could be implemented such that a L5NIC is present on either one of the respective sides. Both client and server begin the operation by sending a "tl_open" message for the purpose of opening a connection to the transport accelerator provider (TAP). A comparison between "tl_" constructs and FreeBSD is shown in Fig. 11. On the server side, "tl_open" is followed by sending "tl_listen" and "tl_accept" to TAP. At this stage the server is ready to handle requests to open a connection. On the client side, following "tl_open," the primitives "tl_transfer_in", telling the TAP to push the connection to a certain accelerator, and "tl_connect", telling the TAP to trigger a connection setup of a 3-way handshake, are sent to TAP. TAP responds with a "TRANS_IN" request to the TCP handler (TCP FW) and TCP FW responds with a "SYN" signal sent through the network to the TCP FW of L5NIC of the server side. The TCP FW sends a "PACKET_NOTIFY" to the TAP of the server to begin a new connection and in response, ULP of server responds with "tl_listen_cb", which is a call back that tells the ULP that there is a new request for a connection setup, "tl_transfer_in", which tells the TAP to push the connection on a specific L5NIC, and

"tl_confirm", which causes the TAP to complete the connection setup handshake. After this process the server is ready to accept data from the connection. The server's TAP initiates a "TRANS_IN" and the TCP FW of the server's L5NIC responds with a SYN/ACK signal to the network directed at the requesting client.

[60] The TCP FW upon receiving the SYN/ACK signal sends back, to the server, an ACK signal and causes a notify message to the ULP FW making it ready to open a direct ULP to ULP FW connection. In addition a "CONN_NOTIFY" message is sent from the TCP FW of the L5NIC of the client to the host TAP which in turn responds with a "tl_connected_cb" message to the ULP of the client, thereby facilitating the direct connection between the host ULP and the ULP FW. As a result acceleration of TCP/IP processing is achieved. Similarly, in response to receiving the ACK signal at the L5NIC of the server, the sequence of notification and connection occurs. As a result there is a direct link between the server ULP and its respective L5NIC ULP FW, resulting in an acceleration of the TCP/IP processing.

[61] Reference is now made to Fig. 8 where a communication diagram showing the data transfer after a connection is available, from the perspective of the client or server, is shown. Specifically, the capability of an example implementation of an apparatus embodying aspects of the disclosed teachings to operate directly from the ULP to the ULP handler of an L5NIC is depicted. It should be noted that this diagram is applicable at a time after a connection was established as was described in Fig. 7 above. Specifically the host may generate a connection_send request aiming at the sending of data to, for example, a server, via the network. The data is also sent to the L5NIC as a data block and it is the responsibility of the L5NIC to handle splitting the data into packets. Packets of data are sent out and

acknowledgments of packets received by the remote location are received without any further intervention by the host. All such activity is handled by the L5NIC without intervention by the host. Upon receiving acknowledgment of the last packet being received by the remote location, a notification is sent to the ULP FW of the L5NIC, which in return sends a connection_send_notify message to the host ULP, to confirm that the data was received by the remote location. Therefore it is clear that the host was relieved from the need to handle the data packet send operation resulting in overall performance acceleration of this activity.

[62] Referring to Fig. 9, a communication diagram of a closure of a client to server connection is shown. The host initiates a disconnect by sending the message "tl_disconnect" to the TAP. TAP then sends a "DISCONNECT" message to the TCP FW which responds with a "notify" message to the ULP FW. Any data that needs to be sent is sent over the network followed by a "FIN" signal. The FIN signal is received by the TCP FW of the server's L5NIC which responds with an ACK signal back to the client side, and a "notify" signal to the ULP FW of the server's L5NIC. The ULP FW sends an end of file (EOF) notification to the ULP which responds with a "tl_disconnect" to the TAP. TAP responds with a "DISCONNECT" message to the TCP FW. Any remaining data is sent to the network and received by the client, followed by a "FIN" signal sent from the L5NIC of the server to the network. The "FIN" message is received by the L5NIC of the client side and responds with an "ACK message to the server, in addition to notification to the ULP FW as well as a "CONN_NOTIFY: to the TAP of the server. TAP responds with a "tl_disconnect_cb to the ULP which concludes with a "tl_close" to the TAP to clean all the resources associated with the connection on the host side. Similarly, upon receipt of an "ACK"

message by the L5NIC of the server, a similar sequence of events takes place causing the connection to terminate. This communication model where a L5NIC is implemented on only one of the client or server sides can be implemented. While a closure of connection by a client to a server is described hereinabove, the operation for the closure of a connection initiated by a server to a client is also within the scope of the disclosed teachings.

[63] Other modifications and variations to the invention will be apparent to those skilled in the art from the foregoing disclosure and teachings. Thus, while only certain embodiments of the invention have been specifically described herein, it will be apparent that numerous modifications may be made thereto without departing from the spirit and scope of the invention.